Transforming Ontology Axioms to Information Processing Rules – An MDA Based Approach

Diana Kalibatiene, Olegas Vasilecas, Giancarlo Guizzardi

Presented by Olegas Vasilecas
Information Systems Research Laboratory
Vilnius Gediminas Technical University
Lithuania
This work is supported by Lithuanian State Science and Studies Foundation according to High Technology Development Program Project *“Business Rules Solutions for Information Systems Development (VeTIS)”* Reg. No. B-07042
Outline

• Introduction (Main Problem, Motivation, Solution)
• Transforming Ontology Axioms to Information Processing Rules
• Transforming PAL Constraints to OCL Constraints
• An Example of Transforming PAL to OCL
• Conclusions and future works
Introduction

• The knowledge-based information systems development using the domain ontology is the hot topic nowadays, since the semantic content expressed by ontology can be transformed to information systems artefacts, thereby reducing the costs of conceptual modelling and later implementation.

• In this context, researches are targeted on transforming ontology to conceptual data model because they have some common aspects, i.e., both include concepts, relationships between concepts and rules (in ontology – axioms).

• However, the modelling and implementing of rules, making an important and integral part of each conceptual data model, is not defined enough.
Introduction - problem

• There is no standard language, method or approach for domain rules modelling and implementation:
  – **Languages**: UML with OCL, ORM, KIF, logic, structural languages (OMG specification SBVR).
  – **Methods**: Demuth *et al.* method, Ross method, ...
  – **Tools used**: PowerDesigner, MagicDraw (OCL constraints), ILog, ....
Introduction - problem

• The problems of rules elicitation and implementation:
  – Domain representatives do not or is not able to use any expert systems or programming languages to define rules. Therefore, it is quite difficult to identify rules from the “business speak” and documents used in application domain.
  – Since terms used in a domain might have double meaning, it is necessary to clarify understanding and clearly define terms used in this domain.
  – Domain rules can be implemented in different ways. Therefore, it is important to determine the most suitable.
Introduction - solution

- Domain ontology should be used for the modelling and implementing of application domain rules, since:
  - both domain ontologies and conceptual data models are intended to capture knowledge about a certain subject domain;
  - from a syntactic perspective, in both artefacts, domain knowledge is expressed in a similar way: terms of concepts, their properties and relationships, and rules (in ontology – axioms).
  - ontology axioms (and ontology as a whole) are typically expressed in a formal way. Therefore, they can in principle be transformed to application domain rules automatically.
• An application domain is presented by an application domain ontology with ontology axioms. These axioms are transformed to information processing rules. The information processing rules are transformed to the corresponding executable rules.

• The meta-models of heavyweight ontology with axioms and information processing rules with their mapping follows.
The ontology metamodel
The meta-model of conceptual data model

1. **ConceptualDataModel**
   - name: char
   - constrained: char
   - constrains: char

2. **Entity**
   - name: char
   - description: char
   - associates: Relationship

3. **Relationship**
   - name: char
   - description: char
   - role: char

4. **AppDomRule**
   - name: char
   - description: char
   - content: char

5. **Domain**
   - name: char
   - description: char

6. **Value**
   - name: char
   - data_type: float
   - length: float
   - precision: char

7. **Role**
   - name: char
   - minCardinality: Integer
   - maxCardinality: Integer
   - dependency: boolean
   - cardinality: Integer = <0..1>
   - condition: RuleClause

8. **InheritanceRelationship**
   - mutually_exclusive: boolean
   - mutually_inclusive: boolean
   - inherit_all: boolean

9. **RuleClause**
   - name: char
   - default_value: char
   - list_of_values: char

10. **Condition**
    - action: char

11. **Event**
    - occurrence: char

12. **Action**
    - trigger: char

**ONTOSE 2009**
The mapping of meta-classes

The mapping of meta-classes is defined as follows:
- a concept maps to an entity,
- a definition maps to an entity description,
- a property maps to an attribute,
- a relationship maps to a relationship,
- a statement maps to a rule clause,
- an axiom maps to an application domain rule,
- a state maps to an action or a condition, and
- a condition maps to a condition.

For the detailed study of the transforming axioms to information processing rules the PAL, which is used to define axioms in Protégé ontology, and Object Constraint Language (OCL), which is used to define information processing rules in UML class diagrams, is chosen.
Transforming PAL Constraints to OCL Constraints 1

ProtégéOntology := \{CLASS^PO, SLOT^PO, REL^PO, VOC^PO, VALUE^PO, DOMAIN^PO, PALCONST\},

- \textbf{CLASS}^PO = \{class^PO_i | i : N\} is a set of concepts, like a customer.
- \textbf{SLOT}^PO = \{slot^PO_i | i : N\} is a set of slots presenting properties of classes, like customer\_name, and their relationships with other classes, like a slot customer in the class order.
- \textbf{REL}^PO = \{is-a, inverse, has\} is a set of relationships, where is-a - the hierarchical relationship between classes, inverse - the inverse relationship between slots, and has presents slots of a class.
- \textbf{VOC}^PO = \{docum^PO_i | i : N\} is a set of class definitions. Each class has one particular definition.
- \textbf{VALUE}^PO = \{value^PO_i | i : N\} is a set of values in an ontology. A value can be a string, a symbol, a class, etc.
- \textbf{DOMAIN}^PO = \{domain^PO_i | i : N\} is a set of domains, which are sets of possible values (value^PO_i) of slots (slot^PO_i).
Transforming PAL Constraints to OCL Constraints

- \( PALCONST := \{palconst^PO_/ \ i : N \} \) is a set of axioms defined by PAL constraints.
- Each element of \( PALCONST \) is a PAL constraint \( (palconst^PO_/) \) defined by:
  - a PAL-name \( (palname^PO_/) \) or a label of a constraint,
  - a PAL-documentation \( (paldocum^PO_/) \) or a description of a constraint,
  - a PAL-range \( (palrange^PO_/) \), which is a set of local and global variables that appear in the statement of the PAL constraint. A PAL-range \( (palrange^PO_/) \) consists of a set of classes \( (class^PO_/) \) and
  - a PAL-statement \( (palstat^PO_/) \), which consists of a set of statements \( (statement^PO_/) \), which are connected using logical connectives and / or quantifiers. A statement \( (statement^PO_/) \) is composed of a class or a slot associated by a relationship \( (ref^PO_/) \) with a class or a slot or a value \( (value^PO_/) \).
Transforming PAL Constraints to OCL Constraints

- $OCLCONST := \{ oclconst_i \mid i : N \}$ is a set of OCL constraints in a UML class diagram. Each element of $OCLCONST$ is an OCL constraint ($oclconst_i$) defined by a context ($context^{OCL}_i$), a type ($statetype^{OCL}_i$) and an OCL statement ($statement^{OCL}_i$).
Transforming PAL Constraints to OCL Constraints 4

• The transformation of a PAL constraint to the corresponding OCL constraint follows:
  – a PAL-range \((\text{palrange}^\text{PO})\) \(\rightarrow\) the context of an OCL constraint \((\text{context}^{\text{OCL}})\);
  – a PAL-statement \((\text{palstat}^\text{PO})\) \(\rightarrow\) an OCL statement \((\text{statement}^{\text{OCL}})\);
  – a PAL-name \((\text{palname}^\text{PO})\) \(\rightarrow\) the name of an OCL constraint;
  – a PAL-documentation \((\text{paldocum}^\text{PO})\) \(\rightarrow\) comments of an OCL constraint, which are denoted by two dashes (- -).

• The analysis of PAL shows that all PAL-statements correspond to OCL invariants.
An Example of Transforming PAL to OCL

• A prototype of the *PAL OCL transformation* plug-in was developed to carry out the experiment of automatic transformation of PAL constraints to OCL constraints.

• In this prototype it is necessary to specify a file, where OCL constraints will be stored and all PAL constrains will be automatically transformed to OCL constraints.
An Example of Transforming PAL to OCL 1

- The Employee end date should be after the start date

<table>
<thead>
<tr>
<th>PAL constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>(%3APAL-RANGE &quot;(defrange ?Employee :FRAME Employee)\n&quot;)</td>
</tr>
<tr>
<td>(%3APAL-STATEMENT &quot;(forall ?Employee \n (&lt; ( 'start_date' ?Employee) ('end_date' ?Employee)))\n&quot;)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>OCL constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>context Employee inv start_date_before_end_date:</td>
</tr>
<tr>
<td>self.end_date &gt; self.start_date</td>
</tr>
</tbody>
</table>
Conclusions

• The analysis of the related works in the field of knowledge-based information systems development using the domain ontology shows that rules used in application domain are represented in the ontology by axioms.

• In this article, we contribute with an MDA-inspired method for transforming ontology axioms to information processing rules. In order to illustrate the presented method of transforming ontology axioms to information processing rules, we show the transforming of PAL constraints to OCL constraints.
Conclusions 1

- The case study as well as the implemented prototype shows that the suggested method can be implemented and used for the automation of ontology axioms transformation to the information processing rules of an information system.
Thank you!

Questions Please